Unsupervised Learning
Framework of supervised learning

**Given**: Pairs \((x_1, y_1), \ldots, (x_n, y_n)\). Think of \(x\) as input and \(y\) as output.

**Learn**: A function \(f(x)\) that accurately predicts \(y_i \approx f(x_i)\) on this data.

**Goal**: Use the function \(f(x)\) to predict new \(y_0\) given \(x_0\).

**Probabilistic motivation**

If we think of \((x, y)\) as a random variable with joint distribution \(p(x, y)\), then supervised learning seeks to learn the conditional distribution \(p(y|x)\).

This can be done either directly or indirectly:

**Directly**: e.g., with logistic regression where \(p(y|x) = \text{sigmoid function}\)

**Indirectly**: e.g., with a Bayes classifier

\[
y = \arg \max_k p(y = k|x) = \arg \max_k p(x|y = k)p(y = k)
\]
Some motivation

- The Bayes classifier factorizes the joint density as $p(x, y) = p(x|y)p(y)$.
- The joint density can also be written as $p(x, y) = p(y|x)p(x)$.
- *Unsupervised learning* focuses on the term $p(x)$ — learning $p(x|y)$ on a class-specific subset has the same “feel.” What should this be?
- (This implies an underlying classification task, but often there isn’t one.)

Unsupervised learning

*Given*: A data set $x_1, \ldots, x_n$, where $x_i \in \mathcal{X}$, e.g., $\mathcal{X} = \mathbb{R}^d$

*Define*: Some model of the data (probabilistic or non-probabilistic).

*Goal*: Learn structure within the data set as defined by the model.

- Supervised learning has a clear performance metric: accuracy
- Unsupervised learning is often (but not always) more subjective
Overview of second half of course

We will discuss a few types of unsupervised learning approaches in the second half of the course.

**Clustering models**: Learn a partition of data $x_1, \ldots, x_n$ into groups.
- Image segmentation, data quantization, preprocessing for other models

**Matrix factorization**: Learn an underlying dot-product representation.
- User preference modeling, topic modeling

**Sequential models**: Learn a model based on sequential information.
- Learn how to rank objects, target tracking

As will become evident, an unsupervised model can often be interpreted as a supervised model, or very easily turned into one.
Clustering

Problem

- Given data $x_1, \ldots, x_n$, partition it into groups called clusters.
- Find the clusters, given only the data.
- Observations in same group ⇒ “similar,” different groups ⇒ “different.”
- We will set how many clusters we learn.

Cluster assignment representation

For $K$ clusters, encode cluster assignments as an indicator $c \in \{1, \ldots, K\}$,

$$c_i = k \iff x_i \text{ is assigned to cluster } k$$

Clustering feels similar to classification in that we “label” an observation by its cluster assignment. The difference is that there is no ground truth.
THE K-MEANS ALGORITHM
**Clustering and K-means**

**K-means** is the simplest and most fundamental clustering algorithm.

**Input:** $x_1, \ldots, x_n$, where $x \in \mathbb{R}^d$.

**Output:** Vector $c$ of cluster assignments, and $K$ mean vectors $\mu$

- $c = (c_1, \ldots, c_n)$, $c_i \in \{1, \ldots, K\}$
  - If $c_i = c_j = k$, then $x_i$ and $x_j$ are *clustered together* in cluster $k$.

- $\mu = (\mu_1, \ldots, \mu_K)$, $\mu_k \in \mathbb{R}^d$ (same space as $x_i$)
  - Each $\mu_k$ (called a *centroid*) defines a cluster.

As usual, we need to define an *objective function*. We pick one that:

1. Tells us what are good $c$ and $\mu$, and
2. That is easy to optimize.
The K-means objective function can be written as

$$\mu^*, c^* = \arg \min_{\mu, c} \sum_{i=1}^{n} \sum_{k=1}^{K} 1\{c_i = k\} \|x_i - \mu_k\|^2$$

Some observations:

- K-means uses the squared Euclidean distance of $x_i$ to the centroid $\mu_k$.
- It only penalizes the distance of $x_i$ to the centroid it’s assigned to by $c_i$.

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} 1\{c_i = k\} \|x_i - \mu_k\|^2 = \sum_{k=1}^{K} \sum_{i: c_i = k} \|x_i - \mu_k\|^2$$

- The objective function is “non-convex”
  - This means that we can’t actually find the optimal $\mu^*$ and $c^*$.
  - We can only derive an algorithm for finding a local optimum (more later).
**Optimizing the K-means objective**

**Gradient-based optimization**

We can’t optimize the K-means objective function exactly by taking derivatives and setting to zero, so we use an iterative algorithm.

However, the algorithm we will use is different from gradient methods:

\[ w \leftarrow w - \eta \nabla_w \mathcal{L} \quad \text{(gradient descent)} \]

**Recall:** With gradient descent, when we update a parameter “\( w \)” we move in the direction that decreases the objective function, but

- It will almost certainly not move to the best value for that parameter.
- It may not even move to a better value if the step size \( \eta \) is too big.
- We also need the parameter \( w \) to be continuous-valued.
K-means and Coordinate descent

Coordinate descent
We will discuss a new and widely used optimization procedure in the context of $K$-means clustering. We want to minimize the objective function

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{c_i = k\} \|x_i - \mu_k\|^2.$$ 

We split the variables into two unknown sets $\mu$ and $c$. We can’t find their best values \textit{at the same time} to minimize $\mathcal{L}$. However, we will see that

- Fixing $\mu$ we can find the best $c$ exactly.
- Fixing $c$ we can find the best $\mu$ exactly.

This optimization approach is called \textit{coordinate descent}: Hold one set of parameters fixed, and optimize the other set. Then switch which set is fixed.
Coordinate descent (in the context of K-means)

Input: \( x_1, \ldots, x_n \) where \( x_i \in \mathbb{R}^d \). Randomly initialize \( \mu = (\mu_1, \ldots, \mu_K) \).

- Iterate back-and-forth between the following two steps:
  1. Given \( \mu \), find the best value \( c_i \in \{1, \ldots, K\} \) for \( i = 1, \ldots, n \).
  2. Given \( c \), find the best vector \( \mu_k \in \mathbb{R}^d \) for \( k = 1, \ldots, K \).

There’s a circular way of thinking about why we need to iterate:

1. Given a particular \( \mu \), we may be able to find the best \( c \), but once we change \( c \) we can probably find a better \( \mu \).
2. Then find the best \( \mu \) for the new-and-improved \( c \) found in \#1, but now that we’ve changed \( \mu \), there is probably a better \( c \).

We have to iterate because the values of \( \mu \) and \( c \) depend on each other. This happens very frequently in unsupervised models.
**K-means Algorithm: Updating \( c \)**

**Assignment step**

Given \( \mu = (\mu_1, \ldots, \mu_K) \), update \( c = (c_1, \ldots, c_n) \). By rewriting \( L \), we notice the independence of each \( c_i \) given \( \mu \),

\[
L = \left( \sum_{k=1}^{K} 1 \{c_1 = k\} ||x_1 - \mu_k||^2 \right) + \cdots + \left( \sum_{k=1}^{K} 1 \{c_n = k\} ||x_n - \mu_k||^2 \right).
\]

We can minimize \( L \) with respect to each \( c_i \) by minimizing each term above separately. The solution is to assign \( x_i \) to the closest centroid

\[
c_i = \arg\min_k ||x_i - \mu_k||^2.
\]

Because there are only \( K \) options for each \( c_i \), there are no derivatives. Simply calculate all the possible values for \( c_i \) and pick the best (smallest) one.
**K-means algorithm: Updating $\mu$**

**Update step**

Given $c = (c_1, \ldots, c_n)$, update $\mu = (\mu_1, \ldots, \mu_K)$. For a given $c$, we can break $L$ into $K$ clusters defined by $c$ so that each $\mu_i$ is independent.

\[
L = \left( \sum_{i=1}^{N} \mathbb{1}\{c_i = 1\} \|x_i - \mu_1\|^2 \right) + \cdots + \left( \sum_{i=1}^{N} \mathbb{1}\{c_i = K\} \|x_i - \mu_K\|^2 \right).
\]

- sum squared distance of data in cluster #1
- sum squared distance of data in cluster #K

For each $k$, we then optimize. Let $n_k = \sum_{i=1}^{n} \mathbb{1}\{c_i = k\}$. Then

\[
\mu_k = \arg \min_{\mu} \sum_{i=1}^{n} \mathbb{1}\{c_i = k\} \|x_i - \mu\|^2 \quad \rightarrow \quad \mu_k = \frac{1}{n_k} \sum_{i=1}^{n} x_i \mathbb{1}\{c_i = k\}.
\]

That is, $\mu_k$ is the mean of the data assigned to cluster $k$. 
**K-means clustering algorithm**

Given: $x_1, \ldots, x_n$ where each $x \in \mathbb{R}^d$

Goal: Minimize $\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{c_i = k\} \|x_i - \mu_k\|^2$.

1. Randomly initialize $\mu = (\mu_1, \ldots, \mu_K)$.

2. Iterate until $c$ and $\mu$ stop changing

   1. Update each $c_i$:
      
      \[ c_i = \arg \min_k \|x_i - \mu_k\|^2 \]

   2. Update each $\mu_k$: Set
      
      \[ n_k = \sum_{i=1}^{n} \mathbb{1}\{c_i = k\} \quad \text{and} \quad \mu_k = \frac{1}{n_k} \sum_{i=1}^{n} x_i \mathbb{1}\{c_i = k\} \]
A random initialization
K-MEANS ALGORITHM: EXAMPLE RUN

Iteration 1
Assign data to clusters
Iteration 1

Update the centroids
K-MEANS ALGORITHM: EXAMPLE RUN

Iteration 2
Assign data to clusters
K-MEANS ALGORITHM: EXAMPLE RUN

Iteration 2
Update the centroids
Iteration 3

Assign data to clusters
K-MEANS ALGORITHM: EXAMPLE RUN

Iteration 3
Update the centroids
K-MEANS ALGORITHM: EXAMPLE RUN

Iteration 4
Assign data to clusters
K-MEANS ALGORITHM: EXAMPLE RUN

Iteration 4
Update the centroids
CONVERGENCE OF K-MEANS

Objective function after
- the “assignment” step (blue: corresponding to $c$), and
- the “update” step (red: corresponding to $\mu$).
Convergence of K-means

The outline of why this convergences is straightforward:

1. Every update to $c_i$ or $\mu_k$ decreases $\mathcal{L}$ compared to the previous value.
2. Therefore, $\mathcal{L}$ is monotonically decreasing.
3. $\mathcal{L} \geq 0$, so Step 1 converges to some point (but probably not to 0).

When $c$ stops changing, the algorithm has converged to a local optimal solution. This is a result of $\mathcal{L}$ not being convex.

Non-convexity means that different initializations will give different results:

- Often the results will be similar in quality, but no guarantees.
- In practice, the algorithm can be run multiple times with different initializations. Then use the result with the lowest $\mathcal{L}$. 
**Selecting $K$**

We don’t know how many clusters there are, but selecting $K$ is tricky. The K-means objective function decreases as $K$ increases,

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{c_i = k\} \|x_i - \mu_k\|^2.$$  

For example, if $K = n$ then let $\mu_k = x_k$ and as a result $\mathcal{L} = 0$.

Methods for choosing $K$ include:

- Using advanced knowledge. e.g., if you want to split a set of tasks among $K$ people, then you already know $K$.

- Looking at the *relative* decrease in $\mathcal{L}$. If $K^*$ is best, then increasing $K$ when $K \leq K^*$ should decrease $\mathcal{L}$ much more than when $K > K^*$.

- Often the K-means result is part of a larger application. The main application may start to perform worse even though $\mathcal{L}$ is decreasing.

- More advanced modeling techniques exist that address this issue.
**TWO APPLICATIONS OF K-MEANS**

**Lossy data compression**

- **Approach:** Vectorize $2 \times 2$ patches from an image (so data is $x \in \mathbb{R}^4$) and cluster them with K-means. Replace each patch with its assigned centroid.

  - (left) Original $1024 \times 1024$ image requiring 8 bits/pixel (1MB total)
  - (middle) Approximation using 200 clusters (requires 239KB storage)
  - (right) Approximation using 4 clusters (requires 62KB storage)

**Data preprocessing (side comment)**

K-means is also very useful for *discretizing* data as a preprocessing step. This allows us to recast a continuous-valued problem as a discrete one.
EXTENSIONS: K-MEDOIDS

Algorithm: K-medoids clustering

Input: Data $x_1, \ldots, x_n$ and distance measure $D(x, \mu)$. Randomly initialize $\mu$.

- Iterate until $c$ is no longer changing
  1. For each $c_i$: Set
     $$c_i = \arg \min_k D(x_i, \mu_k)$$
  2. For each $\mu_k$: Set
     $$\mu_k = \arg \min_\mu \sum_{i: c_i = k} D(x_i, \mu)$$

Comment: Step #2 may require an algorithm.

K-medoids is a straightforward extension of K-means where the distance measure isn’t the squared error. That is,

- K-means uses $D(x, \mu) = \|x - \mu\|^2$.
- Could set $D(x, \mu) = \|x - \mu\|_1$, which would be more robust to outliers.
- If $x \not\in \mathbb{R}^d$, we could define $D(x, \mu)$ to be more complex.